Attorney's Docket No.
014208.1483

PATENT APPLICATION

ELECTRONIC FILE MANAGEMENT

RELATED APPLICATION

This application claims benefit under 35 U.S.C. §119 of United States provisional application serial number 60/340,336 entitled "Electronic File Management," which was filed on December 13, 2001.

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to data management and more particularly to electronic file management.

BACKGROUND OF THE INVENTION

Electronic information management often involves large amounts of data and complex data structures. Users often work with large assemblies that are managed by a central server machine. Large assemblies may comprise several thousand individual part files having links between files. Opening these files directly from the server may be inefficient, especially when users are accessing the files over a network. Additionally, users often move or rename files, making their links to other files invalid. Consequently, subsequent users have difficulty finding the moved or renamed files. A resulting problem is that working with assemblies having multiple individual part files that are managed at a server may be inefficient and difficult for users.

SUMMARY OF THE INVENTION

According to one embodiment of the invention, a method of accessing, by a client, one or more files residing in a server includes requesting, by the client, downloading of a selected file residing in the server. The selected file is associated with at least one associated file. The method also includes initiating downloading of the selected file and automatically determining the identify of, and initiating downloading of, the at least one associated file in response to requesting downloading of the selected file. The method also includes initiating storing, in a memory associated with the client, of the selected file and the at least one associated file under respective local identifiers.

According to another embodiment of the invention, a system includes a server having a document manager stored in the server. The document manager is operable to maintain a respective profile for each of a plurality of files. Each profile includes respective identifications of associated files associated with the file. The system also includes one or more clients associated with the server. Each of the one or more clients has access to at least one computer-readable medium comprising a software program. The software program is operable to request downloading of a selected file residing in the server. The selected file is associated with at least one associated file. The software is also operable to initiate downloading of the selected file and automatically determine the identity of, and initiate downloading of, the at least one associated file in response to the request. The software is also operable to initiate storing, in a memory associated with the client, of the selected file and the at least one associated file under respective local identifiers.

Some embodiments of the invention provide numerous technical advantages. Some embodiments may benefit from some, none, or all of these advantages. For example, according to one embodiment, files are more quickly accessed and easier to work with because the file and any associated files are automatically downloaded into a memory that is associated with the client. Such a method makes it unnecessary for the client to access any associated files, individually or in groups, from the server after the download. According to another embodiment, relocated or renamed files can be found in the server, making it easier for multiple users to access the same data on the server.

Other technical advantages may be readily ascertained by one of skill in the art.

BRIEF DESCRIPTION OF THE DRAWINGS

Reference is now made to the following description taken in conjunction with the accompanying drawings, wherein like reference numbers represent like parts, in which:

FIGURE 1A is a block diagram illustrating an embodiment of a system for managing electronic files;

FIGURE 1B is a schematic diagram illustrating an example file and its related files stored in the system of FIGURE 1A;

FIGURE 1C is a schematic diagram illustrating an example profile associated with the example file illustrated in FIGURE 1B;

FIGURE 1D is a schematic diagram illustrating additional details of the profile illustrated in FIGURE 1C;

FIGURE 2 is a flowchart illustrating an embodiment of a method of preparing files for storage in a server;

FIGURE 3 is a flowchart illustrating an embodiment of a method of managing electronic files; and

FIGURE 4 is a flowchart illustrating further details of a step of identifying associated files in the method of FIGURE 3.

DETAILED DESCRIPTION OF EXAMPLE EMBODIMENTS OF THE INVENTION

Example embodiments of the invention are best understood by referring to FIGURES 1A through 4 of the drawings, like numerals being used for like and corresponding parts of the various drawings.

FIGURE 1A is a block diagram of a system 10 according to the teachings of the present invention. System 10 includes a client 14 that is associated with a server 18 by a link 22. Client 14 may be any device that is capable of managing, generating, or storing data, or client 14 may perform other functions related to any data. One example of client 14 is a computer executing suitable client software. Server 18 may be any device that is capable of managing data and that allows at least one client 14 to access data stored in server 18. Link 22 may comprise a medium capable of transporting data between endpoints, such as client 14 and server 18. System 10 may include a plurality of clients 14; however, only one client 14 is shown for clarity of illustration.

Client 14 includes, in the illustrated embodiment, a processor 32, a memory 28, a storage medium 30, an input device 36, and an output device 40. Processor 32 may be any device operable to process data and execute instructions. An example of processor 32 is the Pentium™ processor available from Intel Corporation; however, other processors may be used. Processor 32 is coupled to link 22. Input device 36, output device 40, memory 28, and storage medium 30 are coupled to processor 32. Memory 28 may be Read Only Memory, Random Access Memory, or may be a removeable medium such as a floppy disk.

Software program 26 may be any instruction or set of instructions that, when executed by processor 32 of client 14, is operable to transmit, receive, generate, copy, or serve other functions that are related to data. Examples of software program 26 are word processing programs, computer-aided drafting programs such as Solid Edge™ available from Unigraphics Solutions, or other commercial or non-commercial programs. Software program 26 may be a part of an application program such as a drawing package. In the example shown in FIGURE 1A, software program 26 resides in memory 28, but software program 26 may also reside in storage medium 30.

Storage medium 30 may be any media that is capable of storing data. An example of storage medium 30 is a conventional hard drive, Compact Disc Read Only memory, Compact Disc Rewritable memory, or other types of electronic data storage. Files 34 reside in storage medium 30 in this embodiment; however, files 34 may also be stored in memory 28. Files 34 may have been generated by client 14 and/or downloaded from server 18. Files 34 may be associated with each other in various ways. Example associations between files 34 are described in conjunction with FIGURE 1B. Storage medium 30 may also store a list 46 describing associations between a given file 34 and its related files, as described in greater detail below. Although only one list 46 is shown, a separate list 46 may be stored in client 14 for each file 34. List 46 may be generated by software program 26. List 46 may alternatively be stored in memory 28.

Server 18 includes storage medium 52 that stores files 56. Files 56 represent versions of files 34 stored on client 14 that may be accessed by a plurality of clients 56. Files 34 are local versions of files 56 that may be modified and then stored as files 56 on server 18. In one embodiment, files 56 may be managed by a document manager 60. In one embodiment, document manager 60 manages files 56 by maintaining an appropriate file structure, indexing any metadata associated with any of files 56, and accounting for files 56 using identifiers, such as a Uniform Resource Locator ("URL"). Metadata refers to a description of data. In one embodiment, document manager 60 may be a web-based portal, such as Microsoft SharePoint™. However, other types of document managers may be used.

FIGURE 1B illustrates an example of the structure of files 34. The illustration of FIGURE 1B may also illustrate an example of the structure of files 56 because files 56 are files 34 that were transferred from client 14. To avoid redundancy of explanation, FIGURE 1B is described using only files 34.

In one embodiment, files 34 may be assemblies generated by software program 26, which may be a drawing package such as Solid Edge™. In this example, file 34A is designated as a "selected file." A "selected file" refers to one of files 34 that is designated for a data management action, such as being opened, uploaded and/or downloaded. In that sense, any one of files 34 may be a selected file at some point in time. For example, file 34A may be the selected file because file 34A is selected to be downloaded by client 14.

Selected file 34A may need to use or access one or more of the other files 34. These files that selected file 34A directly uses are referred to herein as "first generation" descendants. For example, the individual part files of a drawing file created by a drawing package such as Solid Edge™ may be categorized into multiple generations of files; the individual part files used directly by the drawing file are first generation descendants. The first generation descendants in this example are files 34B, 34C, and 34D. Each of the first generation descendants, in turn, may directly use additional files. Files used by a first generation descendant file are referred to herein as second generation files. The second generation files in this example are files 34E, 34F, and 34G. File 34B directly uses second generation files 34E and 34F. File 34C directly uses second generation file 34G. File 34D uses no second generation file. A third generation of descendants in this example is represented by files 34H and 34I, both of which are directly used only by file 34G. The generations of descendants may continue depending on the needs of the selected file.

Although files 34B through 34I are categorized into multiple generations, all of files 34B through 34I are referred to as associated files of file 34A because files 34B through 34I are descendants of file 34A. A descendant of a selected file is a file that will be used by the selected file or is used by another descendant of the selected file. Files 34B, 34C, and 34D are referred to as immediately associated files of file 34A because file 34A directly uses these files without going through an intermediate file. Once files 34B, 34C, and 34D are selected for access and/or downloading, each of files

34B, 34C, and 34D may be referred to as a selected file. As the selected files, files 34B, 34C, and 34D each may have immediately associated files among the second generation descendants. For example, file 34E and file 34F are immediately associated files of file 34B because from file 34B's point of view, file 34B must access file 34E and file 34F to properly support file 34A. File 34C has the associated files of files 34G, 34H, and 34I, but only file 34G is an immediately associated file because from file 34C's point of view, access to file 34G is necessary to properly support the function of file 34C. File 34D has no immediately associated file.

In a conventional data management system, client 14 executing software program 26 may interact with server 18 over link 22 to upload, store, and/or download one or more files 34. For example, client 14 may generate file 34A and associated files 34B through 34I. Client 14 generates an identifier for each of files 34, and uploads files 34, along with any relevant metadata associated with each of files 34 to server 18 and stores the fiels as files 56. Document manager 60 of server 18, in turn, manages files 56 and indexes the respective metadata. Because each of files 56 has a corresponding one of files 34, in this example, files 56 include the same file structure as files 34, illustrated in FIGURE 1B. To avoid redundancy of explanation, files 56 are referred to in the below example by reference to their corresponding files 34. When client 14 wishes to download file 34A, client 14 sends a request for file 34A. Document manager 60 locates file 34A and transmits file 34A to client 14. Client 14 receives file 34A, but does not automatically obtain the file that file 34A uses, either directly or indirectly, namely files 34B through 34I. But these files are needed to use file 34A. Obtaining the multiple levels of descendant files associated with file 34A may be time consuming, cumbersome, and may require significant user interaction. Furthermore, locating certain ones of associated files 34B through 34I may be difficult if another user accessing those files renames or relocates any of them.

According to the teachings of the invention, an apparatus, a method, and a system are provided that improve the efficiency of using files 34. In one embodiment, efficiency may be improved by generating a profile for each of files 34 that facilitates downloading, all at once, any associated files necessary to use a particular one of files 34. This is advantageous because having all of the files associated with a particular file stored locally in client 14 allows client 14 to work more efficiently with files 34. Furthermore, renamed or relocated files 34 may be located using a profile associated with the renamed or relocated files. Additional details of example embodiments of the apparatus, the system, and the method are described in greater detail below in conjunction with FIGURES 1C through 4.

FIGURE 1C illustrates one embodiment of a profile 38 and a status file 42. A separate profile 38 and status file 42 may be stored for each file 34, in one embodiment. Profile 38 and status file 42 are not explicitly shown in FIGURES 1A and 1B. In one embodiment, profile 38 for any given file 34 may identify files that are immediately associated with the file. For example, for file 34A, profile 38 lists files 34B through 34D as immediately associated files of file 34A. A profile for file 34B (not explicitly shown) may in turn list files 34E and 34F as being immediately associated with file 34B. In another embodiment, profile 38 may identify all of associated files 34B through 34I for file 34A. Files 34 may be identified by profile 38 by any type of identifier, including a URL (as shown in FIGURE 1D) and a globally unique identifier. The globally unique identifier is a unique identifier that is associated with each of files 34 that does not change when the file is renamed or relocated in server 18. Document manager 60, such as Microsoft SharePoint™, may index globally unique identifiers for rapid searching. Other indexable information pertaining to each of files 34 may also be listed in profile 38. In one embodiment, there may be more than one profile 38 for each file 34. For example, one profile 38 of file 34A may identify files 34B through 34D by their respective Uniform Resource Locators, while another profile of file 34A may identify files 34B through 34D by their respective globally unique identifiers. Listing associated files, immediate or otherwise, in profile 38 facilitates identifying all files used by file 34A, which facilitates downloading those files for use by software program 26.

Status file 42 may contain information such as the time of download, check out and check in status, and status of modification of any given file. Each of files 34 may have a status file 42 assigned to it. Status file 42 is generated by software 26, but could be generated by other components, such as document manager 60. Status file 42 may be a cookie file. Having a status file 42 associated with each of files 34 is advantageous because the information pertaining to each of files 34 in status file 42 may be used to facilitate updating files 34 for transferring back to server 18.

In operation, system 10 allows management of files 34 and files 56 by generating and examining profile 38 associated with each of files 34 and 56. Software program 26 may generate file 34A and prepare it for transfer to server 18, making file 34A available to all clients 14. In generating file 34A, in one embodiment, software program 26 also creates files 34B through 34I, which are necessary to present or use the information in file 34A. For each of files 34, software program 26 generates at least one profile 38. Once a respective profile 38 for each file 34 is prepared, software program 26 transmits files 34 to document manager 60 of server 18. In turn, document manager 60 receives the transmission and stores files 34 and respective profiles 38 in storage medium 52 as files 56. Generating profiles 38 identifying the files needed to use any given file 34 is advantageous because those associated files may be downloaded all at once and stored locally on client 14. Client 14 may either identify all associated files at once or alternatively, recursively examine each of profiles 38 associated with each of the immediately associated files until all associated files (descendants) are identified and downloaded. Further details of examining profiles 38 are described below in conjunction with FIGURES 3 and 4. Once files 56 are stored in server

18 with respective profiles (profiles not explicitly shown in FIGURE 1A), files 56 are ready to be downloaded by client 14, when needed again.

At a user's command, software program 26 on client 14 requests download of one of files 56. Software program 26 may send the request for download over link 22 to document manager 60 of server 18. Upon receiving the request, document manager 60 locates and transmits the file 56 and its associated profile 38 to client 14 to be stored in storage medium 30.

In one embodiment, software program 26 examines profile 38 of the downloaded file to identify immediately associated files (those files directly used by the downloaded file). Then software program 26 creates a list 46 that identifies the immediately associated files of the downloaded file. For example, for the example where downloaded file 56 corresponds to file 34A, list 46 may identify the files 56 corresponding to files 34B through 34D as immediately associated files. Software program 26 then sends a request to document manager 60 to examine the respective profiles 38 of the immediately associated files 56 corresponding files 34B through 34D. Upon examination, software program 26 identifies the immediately associated files of the files 56 corresponding to file 34B, file 34C and file 34D and stores their respective identifiers on list 46. Once the immediately associated files in one level of descendants are determined, software program 26 identifies the immediately associated files in the next level of descendants in list 46. This process continues until list 46 identifies all of the associated files or descendants of the downloaded file. Software program 26 then uses list 46 to request download of all associated files identified on list 46. Once all associated files of file 34 are downloaded, they are stored in storage medium 30.

In another embodiment, where profile 38 lists all associated files or descendants of file 34A, software program 26 identifies all associated files by examining profile 38 and requests download of all associated files from server 18. One of skill in the art may determine other procedures to determine identities of all associated files of a particular file using profile 38.

FIGURE 2 is a flowchart illustrating an embodiment of a method 78 of preparing files for storage in server 18. In one embodiment, method 78 may be implemented by system 10 shown in FIGURE 1. The file structure shown in FIGURE 1B is used as a representative example to describe method 78. Method 78 starts at step 80. At step 84, file 34A is designated as a selected file for transfer to server 18. In one embodiment, file 34A may have been generated by software program 26. Once file 34A has been designated as the selected file, in one embodiment, profile 38 of file 34A identifies files that are immediately associated with file 34A at step 88. Examples of the immediately associated files of file 34A are files 34B through 34D (shown in FIGURE 1B). At step 92, profile 38 for file 34A is generated; in one embodiment, profile 38 lists files immediately associated with file 34A. In one embodiment, other information such as a globally unique identifier for each of the immediately associated files may be listed in profile 38.

In another embodiment, profile 38 may identify the immediately associated files using the Uniform Resource Locators.

At step 98, software program 26 determines whether any associated files of file 34A is without a profile 38. Steps 84 through 98 are repeated for each of the files 34B through 34I, so that each profile 38 of each associated file identifies that associated file's immediately associated files. For example, file 34B is designated as the selected file at step 84. Then files 34E and 34F are identified as the immediately associated files of file 34B at step 88. At step 92, profile 38 is generated that lists files 34E and 34F as immediately associated files. At step 98, software 26 determines that there are still other associated files requiring generation of a profile listing its descendants. Thus, steps 84 through 98 of method 78 are repeated again. File 34C is designated as the selected file at step 84. Then file 34G is identified as the only immediately associated file of file 34C at step 88. At step 92, a profile 38 is generated that lists file 34G as being the immediately associated file.

Upon going back to step 84 at step 98 and designating file 34D as the selected file, software 26 recognizes that file 34D has no immediately associated files. As such, in one embodiment, each of the next generation of files are designated as a selected file, and steps 84 through 98 of method 78 are repeated for the remaining associated files until all of the associated files are examined for any immediately associated files. If immediately associated files are found, then the immediately associated files are identified in a profile 38 and associated with the respective file. The end result, in this example, is that a profile 38 of file 34B identifies files 34E and 34F. A profile 38 for file 34C identifies file 34G. A profile 38 of file 34G identifies files 34H and 34I. Each of files 34D, 34H, and 34I has associated with it a profile 38 listing no immediately associated files.

Software 26 may identify all associated files of the selected file at step 88, and not just immediately associated files, and generate a profile 38 identifying all associated files, in one embodiment. In that embodiment, steps 84 through 96 are not repeated because all associated files of file 34A are listed in profile 38.

Then at step 100, file 34A and all of its associated files of file 34B through file 34I are transmitted to server 18 over link 22 for storage as files 56. Method 78 concludes at step 104. Method 78 is advantageous because it allows client 14 to rely on examining the profile 38 for any given file 56 to determine the associated files it uses when downloading that file. Determining the files required by any given file ahead of time allows client 14 to download, all at once, all of the associated files, increasing the efficiency of file access.

FIGURE 3 is a flowchart illustrating a method 110 of accessing, by client 14, files 56 in server 18. In one embodiment, method 110 may be implemented by system 10 shown in FIGURE 1. Files 56 are files 34 that were generated and prepared by client 14 using method 78 and transferred to server 18 for storage as files 56. Because each of files 56 has a corresponding one of files 34, in this example, files 56 include the same file structure as files 34 illustrated in FIGURE 1B. To avoid redundancy of explanation, files 56 are now referred to as files 34 to describe method 110. In addition, individual files of files 56 are now referred to as files 34A through 34I.

Method 110 starts at step 114. At step 118, software program 26 transmits a request to server 18 for downloading one of files 34, such as file 34A, and receives file 34A with an associated profile 38. Then software program 26 identifies files that are associated with file 34A at step 130. In an embodiment in which profile 38 identifies all associated files (files 34B through 34I, in this example), software program 26 initiates download of all associated files at step 134. In one embodiment in which profile 38 identifies only the immediately associated files (files 34B through 34D, in this example), the respective profiles of the immediately associated files, their immediately associated files (in this example, files 34E, 34F, and 34G), and so on, are recursively examined until all associated files of file 34A are identified. Then at step 134, downloading of all associated files is initiated. Further details of that embodiment are discussed in conjunction with FIGURE 4.

At step 138, if one or more of the associated files cannot be found in server 18, then software program 26 initiates a search for the missing files using their respective globally unique identifiers at step 156. Once all associated files are downloaded, in one embodiment, the associated files and the selected file are stored in a local memory under local identifiers at step 142. For example, files 34 may have been stored in server 18 under the following URL format:

HTTP:\server name\work space\folder structure

The URL format above can be modified as the following local identifier:

C:\root directory\server name\work space\folder structure

Storing files 34 in a local memory under local identifiers as shown in the example above allows the user to access files 34 as local files, which improves efficiency of file access.

In one embodiment, software program 26 generates status file 42 at step 158 for each of files 34 and maintains status file 42 in storage medium 30 by updating, information stored in status file 42, such as check out/check in status and time stamp. Once a user finishes using file 34A and all of its associated files 34B through 34I, software program 26 transmits all of files 34 back to server 18 at step 162, along with all the updated information of status file 42. Method 110 concludes at step 146.

This method is advantageous, at least in some embodiments, because client 14 may access file 34A and all of its associated files (file 34B through file 34I) as local files by downloading files 34 at approximately the same time into storage medium 30. Method 110 eliminates the need for software program 26 to access server 18 over link 22 multiple times to download the associated files because all of the associated files are identified first, and subsequently downloaded from server 18 in this embodiment. Method 110 is also advantageous because if one of files 34 has been relocated or renamed, then the globally unique identifiers may be used to find the missing files and update the respective profile 38 and status file 42 to reflect the new location of the missing files. Storing files 34 in storage medium 30 under local identifiers allows software 26 to access files 34 to the user as local files, which improves efficiency of file access.

FIGURE 4 is a flowchart illustrating further details of one embodiment of step 130 of identifying the associated files shown in FIGURE 3. In this embodiment, profile 38 associated with file 34A identifies only the immediately associated files of file 34A (files 34B through 34D in this example). Software program 26 examines profile 38 of the selected file, such as file 34A, at step 170. At step 172, software program 26 determines whether profile 38 lists any immediately associated files, such as files 34B through 34D. If there is one or more immediately associated files, then the identifiers of the immediately associated files are determined at step 174 from profile 38. At step 188, software program 26 adds the identifiers of the immediately associated files to list 46. Then software program 26 repeats steps 170 through 188 for each of the immediately associated files of file 34A until no more immediately associated files can be found. For example, at step 170, software program 26 examines a profile 38 of file 34B. After determining that files 34E and 34F are immediately associated files of 34B at step 172, software program 26 determines the identifiers of files 34E and 34F (which, in this example, are E and F) at step 174. Software 26 then adds the identifiers to list 46. Steps 170 through 188 are repeated again in this manner for file 34C, where a profile 38 for file 34C identifies file 34G as an immediately associated file, determines file 34G's identifier (G, in this example), and adds "G" to list 46. Once list 46 identifies the immediately associated files of all associated files of file 34A, the associated files on list 46 are downloaded at step 134.

Methods and systems described in detail above offer a solution to difficulties related to managing electronic files. One benefit from some embodiments provides quick access to a relevant file and all of the associated files necessary to use the relevant file because all of the necessary files are accessible as local files. Another benefit from some embodiments provides a way to find and download a selected file and all of its associated files even if one or more of them are either renamed or relocated. This may be performed by searching for the missing files using globally unique identifiers.

Although the present invention has been described in detail it should be understood that various changes, substitutions and alterations can be made hereto without departing from the spirit and scope of the invention as defined in the appended claims.